AMENDMENTS TO THE CLAIMS

1. (Currently Amended) A method of a development and build environment for packaged software delivery in a distributed network of nodes, the method comprising the computer-implemented steps of:

compiling source code files into executable file modules;

wherein a module contains an image for a process or a dynamically linked library (DLL);

creating a software package that comprises at least one module;

wherein packages are created based on features/characteristics or purpose;

creating metadata for a first module that includes any module information such as the module's: binary signature, name, directory path, and characteristics;

inserting the module's metadata into the software package;

gathering application program interface (API) dependency information for [[each]] the first module[[;]] wherein [[a]] the first module can provide and use at least one API~~;~~, by

(a) receiving a list of dependent modules for a given process of DLL module;

(b) storing, in the first module's metadata, dependency information for the dependent modules in the list wherein dependency information includes API names and version that the process of DLL depends on;

(c) collecting additional dependency information documented in one or more additional modules specifications wherein, dependency information includes API and versions that the process or DLL depends on; and

(d) storing the additional dependency information in the first module's metadata.

~~collecting dependencies documented in module specifications and placing them into the module's metadata; and~~

~~wherein the dependencies documented in each module lists API names and versions that the process or DLL depends on.~~

2. (Currently Amended) A method as recited in Claim 1,

wherein a linker creates [[a]] the list of dependent modules for a given process or DLL module and places the list in the first module's metadata.

3. (Previously Presented) A method as recited in Claim 1, further comprising the steps of:
creating metadata for each API;
inserting the API metadata into the software package; and
wherein metadata for an API includes, but is not limited to: the API's name and version.

4. (Original) A method as recited in Claim 1, further comprising the step of:
calculating a binary signature for each module and inserting the binary signature into the respective module's metadata; and
wherein each unique version of a module will have a unique binary signature.

5. (Previously Presented) A method as recited in Claim 1, further comprising the steps of:
creating metadata for a package that includes any package information such as the package's: name, build date, and characteristics; and
inserting the package metadata into the package.

6. (Currently Amended) A method of a development and build environment for packaged software delivery in a distributed network of nodes, the method comprising the computer-implemented steps of:
compiling source code files into executable file modules;
wherein a module contains an image for a process or a dynamically linked library (DLL);
creating a software package that comprises at least one module;
wherein a module can provide and use at least one API.
creating metadata for a first module that includes any module information such as the module's: binary signature, name, directory path, and characteristics;
inserting the module's metadata into the software package;
gathering application program interface (API) dependency information for [[each]] the first module[[;]] wherein [[a]] the first module can provide and use at least one API~~;~~, by
(a) receiving a list of dependent modules for a given process of DLL module; and
(b) storing, in the first module's metadata, dependency information for the dependent modules in the list wherein dependency information includes API names and version that the process of DLL depends on;

7. (Currently Amended) A method as recited in Claim 6,
wherein a linker creates [[a]] the list of dependent modules for a given process or DLL module and places the list in the first module's metadata.

8. (Currently Amended) A method as recited in Claim 6, further comprising the step of:
collecting additional dependencies documented in one or more additional module specifications and placing them into the first module's metadata; and
wherein the dependencies documented in each module lists API names and versions that the process or DLL depends on.

9. (Previously Presented) A method as recited in Claim 6, further comprising the steps of:
creating metadata for each API;
inserting the API metadata into the software package; and
wherein metadata for an API includes, but is not limited to: the API's name and version.

10. (Original) A method as recited in Claim 6, further comprising the step of:
calculating a binary signature for each module and inserting the binary signature into the respective module's metadata; and
wherein each unique version of a module will have a unique binary signature.

11. (Original) A method as recited in Claim 6, wherein packages are created based on features/characteristics or purpose.

12. (Previously Presented) A method as recited in Claim 6, further comprising the steps of:
creating metadata for a package that includes any package information such as the package's: name, build date, and characteristics; and
inserting the package metadata into the package.

13. (Currently Amended) An apparatus for a development and build environment for packaged software delivery in a distributed network of nodes, comprising:
means for compiling source code files into executable file modules;
wherein a module contains an image for a process or a dynamically linked library (DLL);

means for creating a software package that comprises at least one module;

means for creating metadata for a first module that includes any module information such as the module's: binary signature, name, directory path, and characteristics;

means for inserting the module's metadata into the software package;

means for gathering application program interface (API) dependency information for [[each]] the first module[[;]] wherein [[a]] the first module can provide and use at least one API~~;~~, by

(a) receiving a list of dependent modules for a given process of DLL module; and

(b) storing, in the first module's metadata, dependency information for the dependent modules in the list wherein dependency information includes API names and version that the process of DLL depends on;

14. (Currently Amended) An apparatus as recited in Claim 13, further comprising:

a linker; and

wherein said linker creates [[a]] the list of dependent modules for a given process or DLL module and places the list in the first module's metadata.

15. (Currently Amended) An apparatus as recited in Claim 13, further comprising:

means for collecting additional dependencies documented in one or more additional module specifications and placing them into the first module's metadata; and

wherein the dependencies documented in each module lists API names and versions that the process or DLL depends on.

16. (Previously Presented) An apparatus as recited in Claim 13, further comprising:

means for creating metadata for each API;

means for inserting the API metadata into the software package; and

wherein metadata for an API includes, but is not limited to: the API's name and version.

17. (Previously Presented) An apparatus as recited in Claim 13, further comprising:

means for calculating a binary signature for each module and inserting the binary signature into the respective module's metadata; and

wherein each unique version of a module will have a unique binary signature.

18. (Original) An apparatus as recited in Claim 13, wherein packages are created based on features/characteristics or purpose.

19. (Previously Presented) An apparatus as recited in Claim 13, further comprising:
means for creating metadata for a package that includes any package information such as the package's: name, build date, and characteristics; and
means for inserting the package metadata into the package.

20. (Currently Amended) A computer-readable medium carrying one or more sequences of instructions for a development and build environment for packaged software delivery in a distributed network of nodes, which instructions, when executed by one or more processors, cause the one or more processors to carry out the steps of:
compiling source code files into executable file modules;
wherein a module contains an image for a process or a dynamically linked library (DLL);
creating a software package that comprises at least one module;
creating metadata for a first module that includes any module information such as the module's: binary signature, name, directory path, and characteristics;
inserting the module's metadata into the software package;
gathering application program interface (API) dependency information for [[each]] the first module[[;]] wherein [[a]] the first module can provide and use at least one API~~;~~, by
(a) receiving a list of dependent modules for a given process of DLL module; and
(b) storing, in the first module's metadata, dependency information for the dependent modules in the list wherein dependency information includes API names and version that the process of DLL depends on;

21. (Currently Amended) A computer-readable medium as recited in Claim 20,
wherein a linker creates [[a]] the list of dependent modules for a given process or DLL module and places the list in the first module's metadata.

22. (Currently Amended) A computer-readable medium as recited in Claim 20, further comprising the step of:

collecting additional dependencies documented in one or more additional module specifications and placing them into the first module's metadata; and

wherein the dependencies documented in each module lists API names and versions that the process or DLL depends on.

23. (Previously Presented) A computer-readable medium as recited in Claim 20, further comprising the steps of:

creating metadata for each API;

inserting the API metadata into the software package; and

wherein metadata for an API includes, but is not limited to: the API's name and version.

24. (Original) A computer-readable medium as recited in Claim 20, further comprising the step of:

calculating a binary signature for each module and inserting the binary signature into the respective module's metadata; and

wherein each unique version of a module will have a unique binary signature.

25. (Original) A computer-readable medium as recited in Claim 20, wherein packages are created based on features/characteristics or purpose.

26. (Previously Presented) A computer-readable medium as recited in Claim 20, further comprising the steps of:

creating metadata for a package that includes any package information such as the package's: name, build date, and characteristics; and

inserting the package metadata into the package.

27. (New) An apparatus running a development and build environment for packaged software delivery in a distributed network of nodes, the apparatus comprising:

a network interface that is coupled to a data network for receiving one or more packet flows therefrom;

a processor;

one or more stored sequences of instructions which, when executed by the processor,

cause the processor to carry out the steps of:

compiling source code files into executable file modules;

wherein a module contains an image for a process or a dynamically linked library (DLL);

creating a software package that comprises at least one module;

wherein packages are created based on features/characteristics or purpose;

creating metadata for a first module that includes any module information such as the module's: binary signature, name, directory path, and characteristics;

inserting the module's metadata into the software package;

gathering application program interface (API) dependency information for the first module wherein the first module can provide and use at least one API, by

(e) receiving a list of dependent modules for a given process of DLL module;

(f) storing, in the first module's metadata, dependency information for the dependent modules in the list wherein dependency information includes API names and version that the process of DLL depends on;

(g) collecting additional dependency information documented in one or more additional modules specifications wherein, dependency information includes API and versions that the process or DLL depends on; and

(h) storing the additional dependency information in the first module's metadata.

28. (New) An apparatus as recited in Claim 27,

wherein a linker creates the list of dependent modules for a given process or DLL module and places the list in the first module's metadata.

29. (New) A method as recited in Claim 27, further comprising the steps of:

creating metadata for each API;

inserting the API metadata into the software package; and

wherein metadata for an API includes, but is not limited to: the API's name and version.

30. (New) An apparatus as recited in Claim 27, further comprising the step of:
calculating a binary signature for each module and inserting the binary signature into the respective module's metadata; and
wherein each unique version of a module will have a unique binary signature.

31. (New) An apparatus as recited in Claim 27, further comprising the steps of:
creating metadata for a package that includes any package information such as the package's: name, build date, and characteristics; and
inserting the package metadata into the package.